Amendments to the claims (this listing replaces all prior versions):

1. (currently amended) A method comprising

maintaining a database that stores data persistently,

accepting tasks from task sources, at least some of the tasks having competing requirements for use of regions of the database, data included in a given region not being available for simultaneous access for writing by more than one of the tasks having competing requirements ~~each of the regions including data that is all either locked or not locked for writing at a given time~~ and data included in different regions being available for simultaneous access for writing by more than one of the tasks having competing requirements,

assigning each of the regions to a respective available processor, each of the regions being assignable to any of the processors,

defining, for each of the tasks, jobs each of which requires write access to a region that is to be accessed by no more than one of the processors, and

distributing the jobs for concurrent execution by the assigned processors.

2. (original) The method of claim 1 in which the stored data includes data items of the database that comprise objects in an object database.

3. (original) The method of claim 1 in which the stored data includes data items that are provided as objects to an object-oriented application.

4. (original) The method of claim 1 in which an object relational broker provides persistent storage of objects for an object-oriented application.

5. (original) The method of claim 1 in which the data is stored in a relational database with object-oriented extensions.

6. (original) The method of claim 1 in which the database comprises files that persistently store the data.

7. (original) The method of claim 1, 2 or 3 in which the number of tasks accepted from task sources is arbitrarily large.

8. (original) The method of claim 1, 2, or 3 in which the number of task sources from which tasks are accepted is arbitrarily large.

9. (original) The method of claim 1, 2, or 3 in which the regions are organized into contention spaces, the number of contention spaces being no less than the number of available processors.

10. (original) The method of claim 9 in which each of the jobs requires write access to data in no more than one of the contention spaces.

11. (original) The method of claim 9 in which the number of contention spaces is equal to the number of available processors.

12. (original) The method of claim 9 in which the organization of regions into contention spaces maximizes the throughput of the available processors in executing the jobs.

13. (original) The method of claim 10 in which the contention spaces are assigned dynamically to processors to maximize the throughput of the available processors.

14. (original) The method of claim 1, 2, or 3 in which the tasks are accepted asynchronously.

15. (original) The method of claim 1, 2, or 3 in which the tasks are accepted concurrently.

16. (original) The method of claim 1, 2, or 3 in which the processors do not use shared memory.

17. (original) The method of claim 1, 2, or 3 in which defining the jobs for each task comprises defining a hierarchy of subtasks in which the lowest level of the hierarchy contains the jobs.

18. (original) The method of claim 1, 2, or 3 in which at least one of the tasks comprises a single job.

19. (original) The method of claim 1, 2, or 3 also including a job generating a task to be performed.

20. (original) The method of claim 1, 2, or 3 in which each of the tasks is completed with a certainty that is at least as high as the certainty that data updated in a requested database transaction is not lost once the transaction is committed.

21. (original) The method of claim 1, 2, or 3 in which the region comprises a single data item.

22. (original) The method of claim 1, 2, or 3 in which the region comprises at least a million data items

23. (currently amended) The method of claim 1, 2, or 3 in which the jobs are executed concurrently ~~without having to wait for release of any write locks on any of the regions~~.

24. (original) The method of claim 9 in which more than one of the contention spaces is associated with one of the processors.

25. (original) The method of claim 24 in which each of the processors comprises a physical processor running at least one process.

26. (original) The method of claim 1, 2, or 3 in which each of the tasks is generated by a user request.

27. (original) The method of claim 9 in which each of the contention spaces is associated with at least two processors one of which executes jobs and the other of which performs administrative functions with respect to the associated contention space.

28. (original) The method of claim 1, 2, or 3 in which the distributing of the jobs includes maintaining a queuing system that has a capacity to receive jobs for execution at an arbitrarily large rate by adding processors proportionately to the number of jobs expected to require execution.

29. (original) The method of claim 28 in which the queuing system includes conceptual rows each of which can receive the jobs.

30. (original) The method of claim 29 in which each of the rows is locked when jobs are being received in the row.

31. (original) The method of claim 30 in which a job can be accepted by the corresponding processor for execution from any of the rows that is not locked.

32. (original) The method of claim 9 in which millions of regions belong to a contention space.

33. (original) The method of claim 1, 2, or 3 in which additional jobs are created in connection with the execution of the jobs.

34. (original) The method of claim 33 in which further jobs are created by the additional jobs.

35. (original) The method of claim 33 in which the creation of the additional jobs is dependent on data read from the database in executing the jobs.

36. (original) The method of claim 9 in which additional jobs are created in connection with the execution of the jobs, and a process running on one of the processors executes the jobs and creates the additional jobs, and in which at least some of the additional jobs are distributed among contention spaces served by other processors.

37. (original) The method of claim 1, 2, or 3 in which the tasks relate to commercial transactions.

38. (original) The method of claim 9 in which each of the jobs is assigned an index associated with the corresponding contention space.

39. (original) The method of claim 38 in which the indexes are used to load balance the jobs among processors.

40. (original) The method of claim 1, 2, or 3 in which the database includes database units that are distributed among different physical locations.

41. (original) The method of claim 1, 2, or 3 in which each of the jobs comprises steps.

42. (original) The method of claim 41 in which execution of a job includes executing a portion of the steps, committing a database transaction representing those steps, and repeating until the job is completed.

43. (original) The method of claim 42 also including, upon a failure to complete any portion of the steps, restarting the execution at the first step of the failed portion with at least the same level of certainty that the job will be completed as the certainty that data written in a requested transaction is not lost once the transaction is committed.

44. (original) The method of claim 31 in which a row is locked only for reading when a processor is accepting jobs from that row.

45. (original) The method of claim 9 in which (1) the distributing of the jobs includes maintaining a queuing system that has a capacity to receive jobs for execution at any arbitrarily large rate, (2) the queuing system includes conceptual rows each of which can receive the jobs, and (3) the queuing system includes conceptual columns associated with respective contention spaces, the queuing system comprising a conceptual matrix of cells at the intersections of the rows and columns, in which each of the cells may be read from or written to without conflicting with reads and writes to other cells.

46. (original) The method of claim 45 in which the rows are associated with sources of jobs and the number of rows is sufficient to permit all of the sources of jobs to load jobs into the queue concurrently.

47. (original) The method of claim 45 in which the number of rows is sufficient to permit jobs to be fetched for execution from all of the columns concurrently.

48. (original) The method of claim 1, 2, or 3 also including synchronizing the executions of synchronization groups of the jobs to ensure correctness of results.

49. (original) The method of claim 48 in which the synchronizing includes assigning to each of the jobs of a synchronization group a tag that identifies them as members of the group.

50. (original) The method of claim 48 in which the synchronizing includes assigning to each of the jobs of a synchronization group a quorum fraction representing the job's proportion of participation in the group.

51. (original) The method of claim 50 in which the jobs are not executed until all of the jobs in the synchronization group are ready for execution by a processor.

52. (currently amended) Apparatus comprising

a database that stores data persistently, and

a job processing mechanism that (1) accepts an arbitrarily large number of tasks asynchronously from an arbitrarily large number of task sources, at least some of the tasks having competing requirements for use of regions of the database, data included in a given region not being available for simultaneous access for writing by more than one of the tasks having competing requirements ~~each of the regions including data that is all either locked or not~~

~~locked for writing at a given time~~ and data included in different regions being available for simultaneous access for writing by more than one of the tasks having competing requirements, (2) organizes the regions into non-conflicting contention spaces each associated with a different available processor, (3) decomposes each of the tasks into jobs each of which requires write access to regions belonging to no more than one of the contention spaces, and (4) distributes the jobs to the corresponding contention spaces for concurrent execution by the associated processors.

53. (original) The apparatus of claim 52 in which the stored data includes data items of the database that comprise objects in an object database.

54. (original) The apparatus of claim 52 in which the stored data includes data items that are provided as objects to an object-oriented application.

55. (original) The method of claim 52 in which an object relational broker provides persistent storage of objects for an object-oriented application.

56. (original) The method of claim 52 in which the data is stored in a relational database with object-oriented extensions.

57. (currently amended) A ~~tangible medium~~ physical article or object bearing instructions to cause a processor to

execute a requiring access to a region of a database that stores data persistently, the job including instructions and pointers to data in the region of the database,

the job that is executed being selected from a contention space of jobs identified by an index, the jobs in the contention space having competing requirements to write into the region of the database, the index distinguishing the contention space from other contention spaces of jobs that do not have competing requirements to write into the region of the database.

58. (currently amended) The ~~method~~ physical article or object of claim 57 in which the stored data includes data items of the database that comprise objects in an object database.

59. (currently amended) The ~~method~~ physical article or object of claim 57 in which the stored data includes data items that are provided as objects to an object-oriented application.

60. (currently amended) The ~~method~~ physical article or object of claim 57 in which an object relational broker provides persistent storage of objects for an object-oriented application.

61. (currently amended) The ~~method~~ physical article or object of claim 57 in which the data is stored in a relational database with object-oriented extensions.

Claims 62 – 67 (cancelled)

68. (previously presented) A method comprising

maintaining a database that stores data persistently and provides a primary level of guarantee that data written in a requested transaction is not lost once the transaction is committed,

accepting tasks from task sources for concurrent execution by multiple processors, at least some of the tasks having conflicting requirements to write into the same region of the database, and

providing a software mechanism that guarantees, as least to the primary level of guarantee, that the tasks will be executed, and will be executed without loss of data and without the occurrence of any actual conflict with respect to the region of the database.

69. (original) The method of claim 68 in which the stored data includes data items of the database that comprise objects in an object database.

70. (original) The method of claim 68 in which the stored data includes data items that are provided as objects to an object-oriented application.

71. (original) The method of claim 70 in which an object relational broker provides persistent storage of objects for an object-oriented application.

72. (original) The method of claim 70 in which the data is stored in a relational database with object-oriented extensions.

73. (original) The method of claim 68, 69, or 70 also including sending to the task source an acknowledgement of acceptance of the task.

74. (original) The method of claim 68, 69, or 70 also including sending to the task source a notification after completion of the accepted task.

75. (original) The method of claim 68 in which the task is decomposed into jobs that are executed by different ones of the multiple processors in a manner that prevents any actual conflict between jobs.

76. (original) The method of claim 74 in which the jobs are subjected to a synchronization mechanism that enables a determination of the completion of a task.

77. (original) The method of claim 76 in which the synchronization mechanism includes a tag that identifies a job as participating in a group of jobs.

78. (original) The method of claim 76 in which the synchronization mechanism includes a quorum fraction that represents the job's proportion of participation in the group.

79. (original) The method of claim 77 also including determining whether the quorum fractions of all of the jobs of a group add to a completed quorum.

80. (original) The method of claim 75 in which the task is notified of completion when all of the jobs have been completed.

81. (original) The method of claim 75 in which the task is assigned to a contention space.

82. (original) The method of claim 75 in which completion notification jobs are assigned for execution in the same contention space as the task.

83. (currently amended) The method of claim 68 in which the database comprises an object-oriented database.